APPLICATION FOR PATENT

Inventors: Moshe Gefen, Shuka Zernovizky, Amir Ban

Title: System and method for enabling non-volatile memory to execute code while operating as a data storage/processing device.

FIELD AND BACKGROUND OF THE INVENTION

The present invention relates to a system for enabling non-volatile memory to execute code while operating as a data storage device.

The primary usage of non-volatile memory is for code execution. In the market of non-volatile code storage memory, flash memory is replacing the ROM various families (ROM ,PROM ,EPROM ,EEPROM) due to its better cost-structure, ease of manufacturing and high densities. Flash is commonly used both as a stand-alone device and as embedded memory. The competition in this market concentrates on condensing the bits of information in a smaller silicon area in order to reduce the cost of the devices. The most common flash type used for code execution is known as NOR Flash. The NOR Flash enables random access to each of its addresses and hence enables to execute code from it. For this reason the NOR Flash is known as an XIP memory, where XIP stands for eXecutable In Place.

While we discussed so far the usage of flash memory for code execution, another emerging market for flash memory is starting to grow and become dominant- the data storage market. Data storage applications require a file system management on the flash memory. Flash memory used for data storage is called a Flash disk and is composed of H/W (flash memory) and a S/W package (file system management, OS interface etc.).

Modern applications usually require flash memory for both code execution and data storage. Today, most architectures use separated devices (or sets of devices) for each functionality. It is very desirable to use the same device (single device) to store both the data and the code of the application. The main benefits are:

reducing real estate requirements, chip count, silicon size and power consumption.

The following scenario illustrates the main problem with this approach:

lets assume that there are two tasks running under the OS in the application.

The first task (T1) is the data storage driver task. It is responsible of storing all the application data on the flash memory. The second task (T2) is some code which is executed from the flash memory (the same flash memory, of course).

The scenario begins with T1 issuing an erase command to some area of the flash memory, as part of the data management requirements. Typical erase time of NOR Flashes is 1 sec. During this period of time (within this 1 sec), OS gives T2 a time slot and T2 starts executing code from the flash memory. At this moment the operation will fail and cause the whole application to fail. The reason is that the flash memory is not available for read operations (e.g., execution of code) while it's busy erasing/ programming another section. The OS and T2 are unaware of the fact that the flash isn't available now. The OS and T2 expect that the code stored on the flash will always be available for execution, but this is not the case. As explained above, there are many cases when the flash is not available for execution of the stored code. In fact – it will be unavailable every time it's busy erasing/programming sections following T1 requests.

Known solutions:

1. Using two devices, one for data storage and the second as code storage (XIP). As

mentioned above, this is the most common architecture that is in use today. See FIGURE 1 for a graphic description of this solution.

This solution has drawbacks of higher real estate requirements, chip count, silicon size and power consumption.

2. Using a single device with multi-bank architecture, which can be simultaneously accessed for read and erase/program. Several flash vendors have started to offer flash devices with multi-bank (usually dual-bank) architecture. With this approach the real estate requirements are reduced and also the chip count is reduced to one. The disadvantage of this solution is the overhead of the silicon (due to the multi-bank design). The estimated cost overhead of this design over a regular design is 30%, so basically one has to pay for the additional functionality with silicon. This solution gains popularity only in real-estate-critical applications, because otherwise it is cost prohibitive. See FIGURE 2 for a graphic illustration of this solution.

3. Using a single device with a special system S/W that controls and schedules all the tasks of the system, for example, in Intel's PSM. This solution uses the S/W commands of suspend and resume of the flash in order to enable the dual functionality of the device. With this solution the problem of unawareness is solved, but the cost is the complicated integration. This requires a solution to be tailored specifically for every CPU and/or OS. The special system S/W is added to the OS and controls and schedules all tasks and interrupts. The time of integration and development of this solution is excessively long since the complexity is high. In addition this is a very intrusive approach, which might suit some niche markets.

There is thus a widely recognized need for, and it would be highly advantageous to have, a system that can enable true simultaneous usage of non-volatile memory for both code execution and data storage.

The present invention provides another approach to solve the problem of one non-volatile (flash) device (or a set of devices) used both for data storage/processing and code execution. The solution enables proper functionality of both usages and in particular will enable execution of code from the flash at any time, including times when the flash is busy erasing/programming some sections.

The present invention is of a hardware application that enables flash memory devices to be both created and operated in an efficient manner, enabling usage of Flash memory for code execution and data storage/processing concurrently.

SUMMARY OF THE INVENTION

According to the present invention there is provided a system for enabling usage of non-volatile memory, and in particular flash memory, for code execution and data storage/processing, comprising:

CPU/Bus/Controller;

Non volatile array;

non volatile device circuitry; and

logic circuit

where said logic circuit is a hardware mechanism that enables supporting of automatic suspend and resume operations.

The present invention provides for designing and implementing an on-chip H/W mechanism that can support an automatic suspend and resume operations. This solution will enable easy integration to any CPU/OS.

The present invention enables a non-volatile memory chip, such as a flash memory chip, to process code execution while it is processing erase/program operations. This is achieved by sensing the operation status of the chip and the CPU/Bus activity, and commanding the flash memory device to suspend and/or resume program/erase operations at appropriate times, so as not to collide with read requests. The system thereby buffers the CPU/Bus/Controller from executing read commands while the chip is processing program/erase operations.

BRIEF DESCRIPTION OF THE DRAWING

The invention is herein described, by way of example only, with reference to the accompanying drawings, wherein:

FIGURE 1 is an illustration of the most common current solution, two separate devices are incorporated into the memory chip for the purposes of code execution and data processing.

FIGURE 2 illustrates an alternative solution to achieving both code execution and data processing in a Flash memory chip, wherein multi-bank architecture is used.

FIGURE 3 illustrates the basic operation of the system according to the present invention.

FIGURE 4 is a flow chart of the basic components of the system according to the present invention.

DESCRIPTION OF THE PREFERRED EMBODIMENT

The present invention is of a system and method for enabling simultaneous usage of code execution and data storage and processing using non-volatile memory chips.

Specifically, the present invention can be used to execute code on flash chips, while concurrently processing stored data on the same chip. This is based on the usage of automatic suspend and automatic resume operations.

The principles and operations of such a system according to the present invention may be better understood with reference to the drawing, and the accompanying descriptions, wherein:

FIGURES 3 and 4 illustrate the basic components and operations of the current invention in its preferred usage. The hardware mechanism of the present invention, which is one logic circuit (or a few circuits), is designed so as to enable automatic suspend and automatic resume of program and/or erase operations in the following manner. For example, consider that there is an active erase/ program command **10** issued to the memory device. The memory device is executing the erase/program operation **15** and at the same time a read request **11** is registered. As opposed to current methods whereby the read request will be unable to be executed, and may crash the system, the present invention temporarily suspends **12** the program/erase operation **15**. When in suspend mode **12**, the CPU/Bus is free to continue with the read requests. The read operation will subsequently be monitored in order to detect a timeout **13** (a predetermined period of

time in which no read operation is done. Upon detection it automatically gives a command to resume operations **14**, allowing the program/erase operation **15** to continue. In greater detail, the system operates as follows:

1. Automatic execution of suspend operation operates on the following conditions:

1.1 The device is busy with erase/program operation **15**.

1.2 A read attempt is being done from the device **11**.

2. Indication of the device entering the suspend state **12** (the time known as suspend latency) is provided with a busy signal **22** (Figure 4). The Busy signal, which is some physical signal to the CPU/Bus, will be used on the platform to signal that the code, stored on the flash chip, is about to be available for execution. The host CPU/Bus/Controller **20** or host Bus **21** uses this signal to hold/retry operation using its standard hold/retry mechanisms, or any other means provided by the CPU/Bus/Controller to prevent a crash due to a failed read attempt.

3. Automatic execution of resume operation **14** upon completion of all the read cycles. Completion will be detected using timeout detection **13** (a predetermined period of time in which no read operation is done). The execution of the resume operation may be commanded based on alternative factors, such as a predetermined time interval or any other chosen method.

Advantages:

First of all, this invention enables using one non-volatile chip, or chip banks acting as unified chips, for both data storage/processing and code execution. By doing this it enables significant reduction of real estate requirements, chip count, silicon size and power consumption. Comparing this invention to the other solution in the market,

Intel PSM (other solutions: a. and b. have a much higher cost) points to the fact that the big advantage here is the easy integration of the flash device (H/W and S/W) to the platform environment (CPU, Bus and OS). With this solution there is no need to interfere with the OS components (e.g. scheduler) and other software ingredients. The OS and all the tasks running under it are totally unaware of the flash memory condition and they can access it regardless of its condition. The only integration required is a simple H/W integration of to allow the CPU, Bus or Controller to hold/retry operations that occur during the suspend latency time. This hardware integration requires the implementation of a regular and common hold/retry mechanism, or any other mechanism existing on the CPU/Bus that can delay execution of a read/fetch cycle. In order to achieve this signaling and enabling dual operation of data storage and code execution on a single chip, the logic circuit needs to be either imbedded in the memory chip or added as an external logic, to facilitate the automatic resume and suspend.

Automatic suspend mechanism:

This section explains the mechanism and implementation of the automatic suspend feature. The automatic suspend logic **26** (Fig 4) is operated when an erase or program operation begins **15** (Fig 3). When detecting one of these operations (erase or program) the automatic suspend logic **26** is triggered. From this moment onwards, the logic waits for a read operation **11** from the device (read operations that requires the device to output real data as opposed to status bits or similar). If the erase/program operation **15** is finished before receiving any read operation **11**, the logic and the chip will both return to the idle state **17**. Identification of the read operation will be based on the regular and normal means that are supplied by the device (e.g. control signals, address

signals, read commands). Upon detection of the read operation **11** the automatic suspend logic **26** executes a process that enters the device into the suspend state **12**. The logic can use existing mechanisms inside the device to do this task (e.g. executing the suspend command which is available in certain devices). In addition – the logic may mark in a certain place (e.g. I/O port or a dedicated register) that the device has entered the automatic suspend state **12**. This marking can be used by the file system management S/W. In addition, the logic will indicate that the device is on its way to the automatic suspend state **12** using an external signal (Busy signal) **22**. This signal can be used by the platform to hold/retry the read operation **11** attempt or any other mechanism in the CPU/Bus that can delay execution of read/fetch cycles. The logic is also responsible of verifying that the device has actually entered the automatic suspend state **12**. After the verification phase – the Busy signal **22** will be turned off (to indicate that the device has entered the automatic suspend state **12**. From this moment onwards the device is ready to perform read requests as required.

Automatic resume mechanism:

The automatic resume logic **27** starts to operate when the device enters the automatic suspend state **12**. The target of this logic is to resume the program/erase operation **15** that was interrupted by the automatic suspend logic **26**. This logic should monitor the read operations done from the device, for example, by using the same techniques as the automatic suspend logic **26**. The logic is responsible to resume the suspended operation. One suggested implementation is to wait for a break in the read operations of the device. When the break is long enough (depending on the application and environment) the logic executes a process which causes the device to resume the

program/erase operation **15** (e.g. executing the resume command which is available in certain devices). The logic contains some mechanism to determine if the break is a real break or just a temporary break (e.g. a timer that counts the no-read-operation time). The logic is also responsible to turn off the mark that shows (e.g. I/O port or a dedicated register) that the device has entered the automatic suspend state **12**.

While the invention has been described with respect to a limited number of embodiments, it will be appreciated that many variations, modifications and other applications of the invention may be made.